Protocoale IoT și integrarea cu servicii Web/Cloud

1. Protocoale de rețea: HTTP/HTTPS, MQTT

Șef Lucrări Dr. Ing. Horia Modran

Contact: horia.modran@unitbv.ro / modranhoria@gmail.com

Tel: 0770171577

Universitatea Transilvania din Brașov

FACULTATEA DE INGINERIE ELECTRICĂ ȘI ȘTIINȚA CALCULATOARELOR

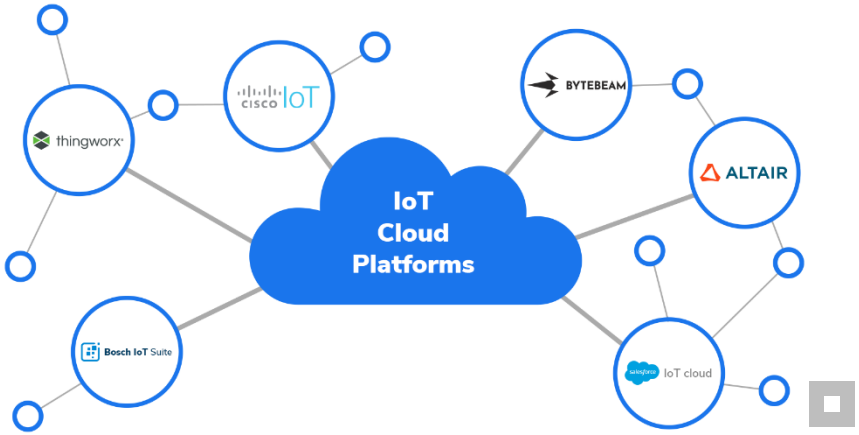
2024 - 2025

Cuprins

- IoT și Web
- Protocolul HTTP
- HTTP Request/Response
- Metode HTTP: GET/POST
- Protocolul MQTT
- Mesaje
- MQTT Client/Broker
- Avantaje/Dezavantaje
- Exemplu implementare

IoT și Web

- **Conectivitate Globală**: prin internet
- **Accesibilitate**: prin aplicații web sau mobile
- **Integrare**: cloud computing, big data și AI
- **Protocole de Comunicație**: HTTP/HTTPS pentru servicii web; MQTT, CoAP pentru IoT (eficiență și consum redus de energie)
- **API-uri REST**: interacțiune aplicațiile web - dispozitivele IoT
- **Cloud Computing**: AWS IoT, Azure IoT Hub oferă stocare, procesare și analiză a datelor

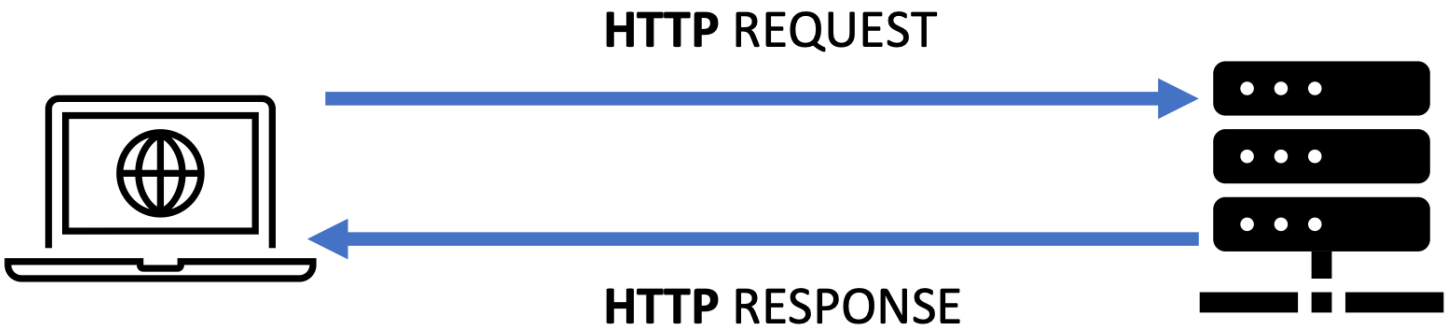


Protocolul HTTP

- HTTP – Hyper Text Transfer Protocol
- Este fundamentul comunicării de date pentru WWW
- Inițiat de Tim Berners-Lee la CERN în 1989
- HTTP/1 a fost finalizat și documentat în 1996
- Specificațiile sale au fost actualizte - HTTP/3 (2022)
- HTTPS este utilizată de 85% dintre site-uri

Protocolul HTTP

- HTTP este protocolul responsabil cu transferul datelor de la serverele web către diverse tipuri de clienți
- Este un protocol peste TCP/IP –> datele nu vor fi deteriorate
- Operează cu date de tip ASCII
- Are la bază conceptele cerere (*request*) și răspuns (*response*)
  - O entitate trimite o cerere și cealaltă oferă un răspuns



Schimbul de date prin HTTP

- HTTP este un protocol *stateless* (nu reține starea sesiunii sau informații de la cererile anterioare)
- Necesită conexiune de transport fiabilă
- HTTP /1 - sunt utilizate conexiuni TCP/IP pe porturile 80 (HTTP) și, respectiv 443 (HTTPS)
- Datele sunt scimbate prin mesaje cerere-răspuns
- Serverul trimite înapoi un mesaj de răspuns HTTP care include antetul (*header*) și corpul (*body*)
- Corpul este de obicei resursa solicitată

Sesiuni HTTP

- Unele aplicații web trebuie să gestioneze sesiunile utilizatorilor, folosind de exemplu cookie-uri HTTP sau variabile ascunse în formularele web
- Cookie – bloc mic de date create de server și salvate de browser pe dispozitiv
- Pentru a începe o sesiune, trebuie efectuată autentificarea
- Pentru a opri o sesiune de utilizator, utilizatorul trebuie să solicite o operație de deconectare

HTTP Request

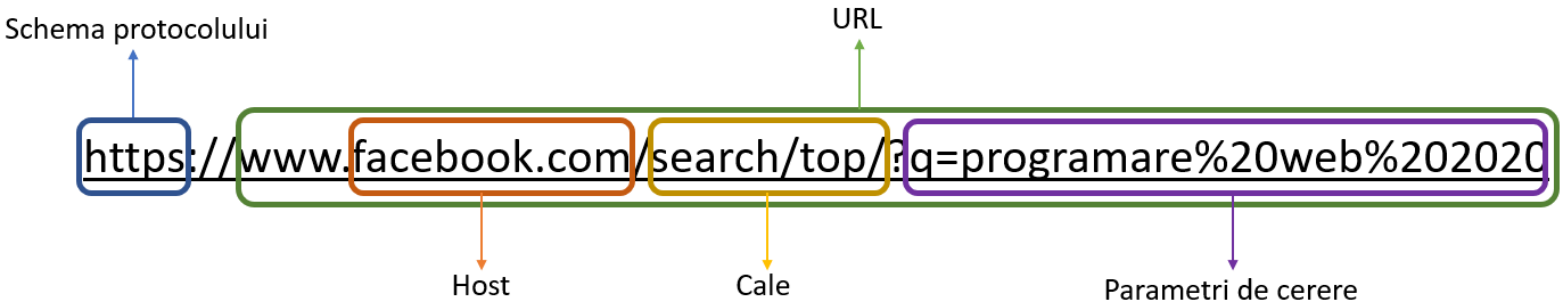
- Mesajele de tip request sunt trimise client către server
- Sintaxa – solicitarea va cuprinde:
  - O linie de start, constând din metodă și URL și protocol

```
GET /images/logo.png HTTP/1.1
```

  - Zero sau mai multe câmpuri de antet (*header fields*)

```
Host: www.example.com
Accept-Language: en
```

  - Opțional: corpul mesajului



Metode HTTP

- HTTP definește metode pentru a indica acțiunea dorită
- Specificația HTTP /1 a definit metodele GET, HEAD, POST
- HTTP /1.1 a adăugat metode noi: PUT, DELETE
- Principalele metode HTTP
  - GET - interogare de resurse
  - POST - crearea de resurse (de obiecei are date atașate)
  - PUT - modificare de resurse (de obiecei are date atașate)
  - DELETE - ștergere de resurse
  - PATCH - modificare parțială a unei resurse (de obiecei are date atașate)

HTTP Response

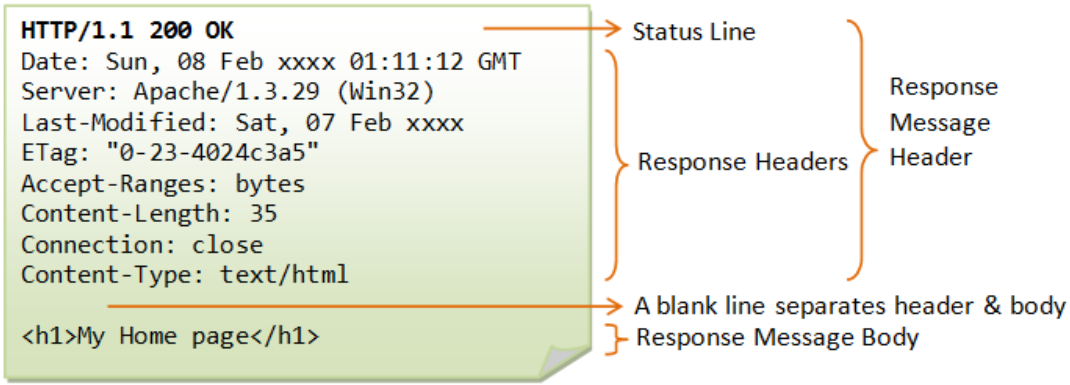
- Un mesaj de răspuns este trimis de la server către client
- Sintaxa – răspunsul va cuprinde:
  - O linie de status - protocolul și codul de stare HTTP

```
HTTP/1.1 200 OK
```

  - Zero sau mai multe câmpuri de antet (*header fields*)

```
Content-Type: text/html
```

  - Opțional: corpul mesajului de răspuns



Status HTTP

- Codul de stare a răspunsului reprezintă rezultatul încercării serverului de a satisface cererea venită de la client
- Este posibil ca unii clienții să nu înțeleagă toate codurile de stare înregistrate, dar trebuie să înțeleagă clasa lor
- Clase de status-uri HTTP
  - 1XX - informational
  - 2XX - successful: 200 OK, 201 Created, 204 No Content
  - 3XX – redirection: 301 Moved Permanently
  - 4XX - client error: 400 Bad Request, 404 Not Found
  - 5XX - server error: 500 Internal Server Error

HTTP GET

- Șirul de interogare (perechi nume/valoare) este trimis în adresa URL: `/demo_form?name1=value1&name2=value2`
- Solicitările GET pot fi stocate în cache și rămân în istoricul browserului
- Request-urile GET pot fi marcate (*bookmark*)
- Nu ar trebui niciodată folosite atunci când se transmit date importante/sensibile
- Solicitările GET au restricții de lungime (2048 caractere)
- Sunt folosite doar pentru a solicita date

HTTP POST

- Datele trimise către server cu POST sunt stocate în corpul cererii HTTP:

```
POST /test/demo_form HTTP/1.1
Host: w3schools.com

name1=value1&name2=value2
```

- Solicitările POST nu sunt niciodată memorate în cache
- Nu rămân în istoricul browserului
- Request-urile POST nu pot fi marcate (*bookmark*)
- Nu se pot transmite direct din browser (doar din client REST)
- Solicitările POST nu au restricții privind lungimea datelor

GET vs POST

	HTTP GET	HTTP POST
Buton back/Reload	Inofensiv	Datele vor fi retrimise (browserul ar trebui să avertizeze utilizatorul că datele sunt pe cale să fie retrimise)
Bookmarked	Da	Nu
Cached	Da	Nu
Tip encoding	application/x-www-form-urlencoded	application/x-www-form-urlencoded sau multipart/form-data. Pentru date binare (fișiere) multipart encoding.
Istoric	Parametrii rămân în istoricul browserului	Parametrii nu rămân în istoricul browserului
Restrictions on data length	Da, la trimiterea datelor, metoda GET adaugă datele la adresa URL; iar lungimea unei adrese URL este limitată la maxim 2048 de caractere.	Fără restricții
Restricții privind lungimea datelor	Sunt permise doar caractere ASCII	Fără restricții. Pot fi transmise de asemenea fișiere
Securitate	GET este mai puțin sigur în comparație cu POST, deoarece datele trimise fac parte din URL. **Nu utilizați niciodată GET când trimiteți parole sau alte informații sensibile!**	POST este mai sigur decât GET, deoarece parametrii nu sunt stocați în istoricul browserului sau în log-urile serverului web.
Vizibilitate	Datele sunt vizibile pentru toată lumea în URL	Datele nu sunt afișate în URL

Idempotență și siguranță

- siguranță (safe) - nu conduce la modificarea stării serverului
- idempotență (idempotent) – cererile identice vor conduce la oferirea aceluiași răspuns (aceeași reprezentare)

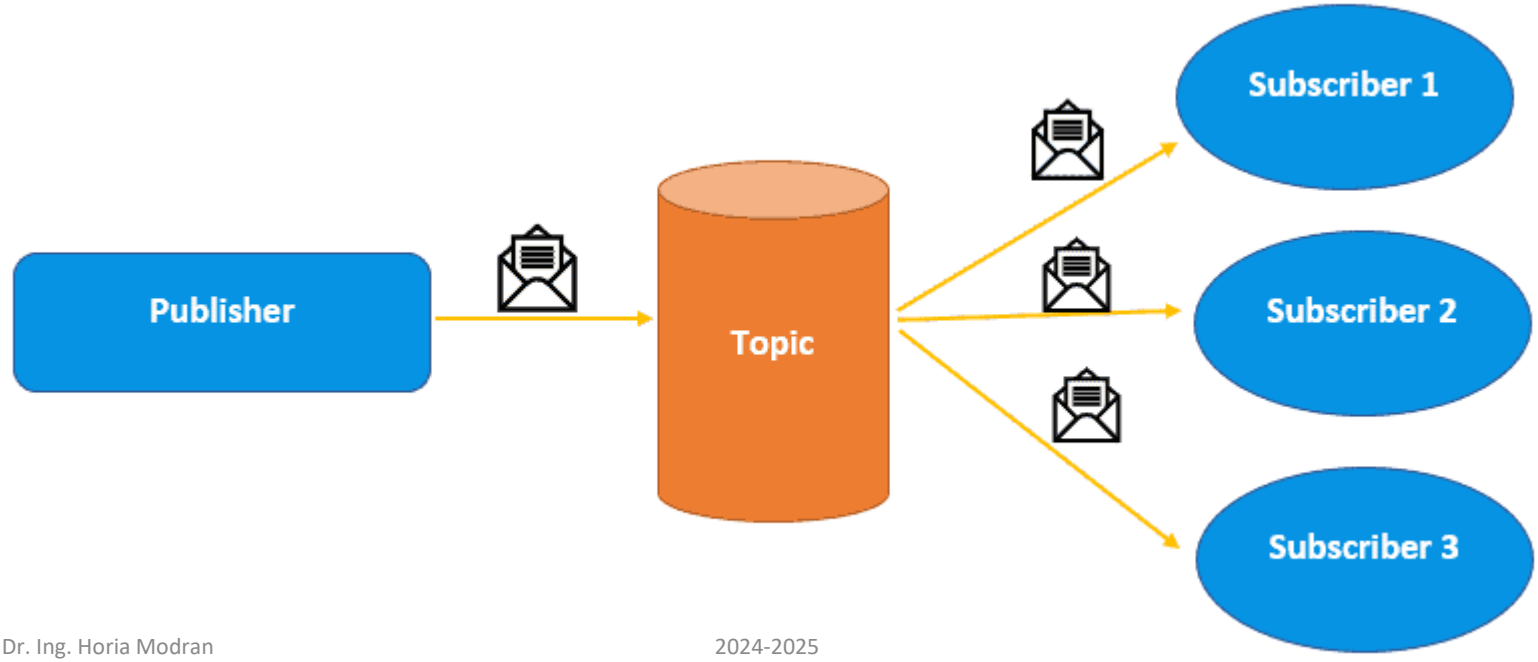
Metoda HTTP	*Idempotent*	*Safe*
GET	✓	✓
POST	X	X
PUT	✓	X
PATCH	X	X
OPTIONS	✓	✓
HEAD	✓	✓
DELETE	✓	X

Protocolul MQTT

- MQTTT = **M**essage **Q**ueuing **T**elemetry **T**ransport
- Protocol de comunicații pentru dispozitivele IoT
- Special conceptul pentru dispozitive cu lățime de bandă mică și latență ridicată
- Lightweight, utilizează modelul *publisher-subscriber*
- Trebuie să ruleze printr-un protocol de transport care oferă conexiuni ordonate, fără pierderi, bidirecționale (TCP/IP)
- Versiunea 5.0 OASIS, lansată în 2019

Modelul Publisher Subscriber

- Model arhitectural utilizat în sistemele distribuite
- Implică 2 componente principale:
  - Publisher – trimite mesaje la sistem pentru un topic
  - Subscriber – se "abonează" la un topic și primește mesaje

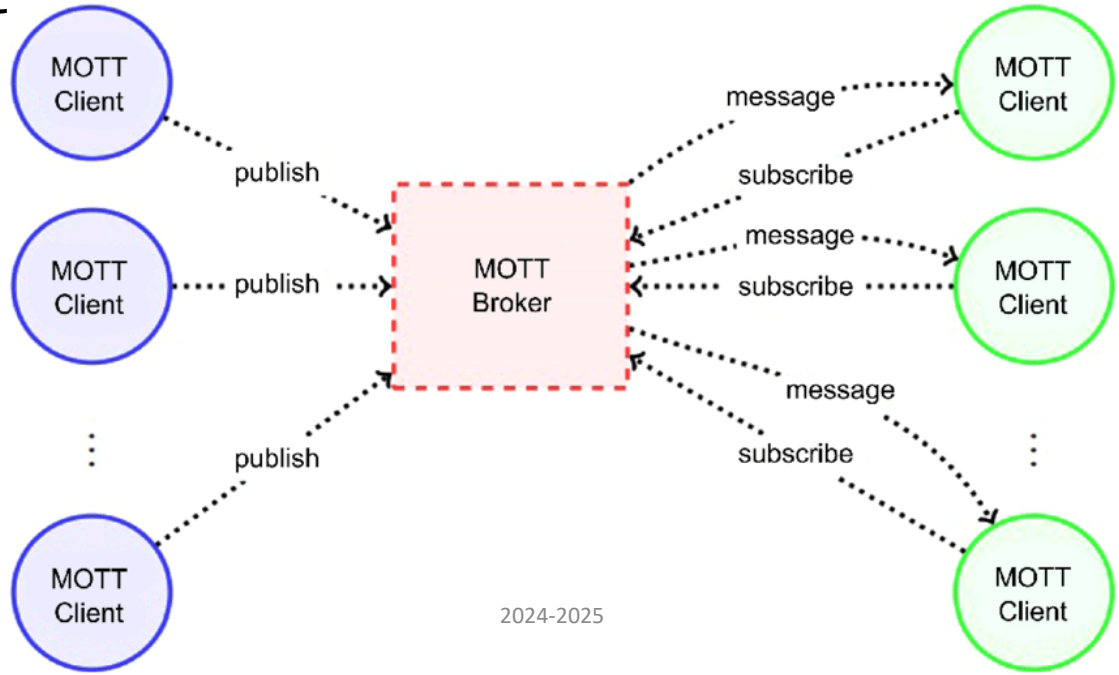


Caracteristici MQTT

- Este un protocol mașină la mașină (asigură comunicarea între dispozitive)
- Conceput ca un protocol de mesagerie simplu și ușor, care utilizează un model de tip publisher/subscriber pentru a face schimb de informații între client și server (broker)
- Nu necesită ca atât clientul, cât și serverul să stabilească o conexiune în același timp
- Oferă o transmisie rapidă și în timp real a datelor, similar cu servicii de mesagerie (WhatsApp)
- Permite clienților să se aboneze la o selecție restrânsă de subiecte, astfel încât să poată primi informațiile necesare

Componente MQTT

- Componentele MQTT
  - Mesaj (*Message*)
  - Client – poate fi *Publisher Client* sau *Subscriber Client*
  - Server (numit și Broker)
  - Topic



Mesaje

- Mesaj - date care sunt transmise de protocolul MQTT în rețea pentru un anumit topic
- Tipuri de mesaje:
  - **Connect** - așteaptă ca o conexiune să fie stabilită cu serverul și creează o legătură între noduri
  - **Disconnect** - așteaptă ca clientul MQTT să termine sarcina curentă și ca sesiunea TCP/IP să se deconecteze
  - **Publish** - revine imediat la firul de execuție al aplicației după transmiterea cererii către clientul MQTT

MQTT Client

- Clienții se abonează la subiecte pentru a publica și a primi mesaje
- Un dispozitiv client: deschide conexiunea de rețea la server, publică mesaje/se abonează la mesaje, se dezabonează la mesajele și închide conexiune de rețea la server
- În MQTT, clientul poate efectua două tipuri de operațiuni:
  - **Publicare**: Când clientul trimite datele către server, atunci numim această operație ca publicare
  - **Abonare**: Când clientul primește datele de la server, atunci numim această operațiune abonare

MQTT Broker

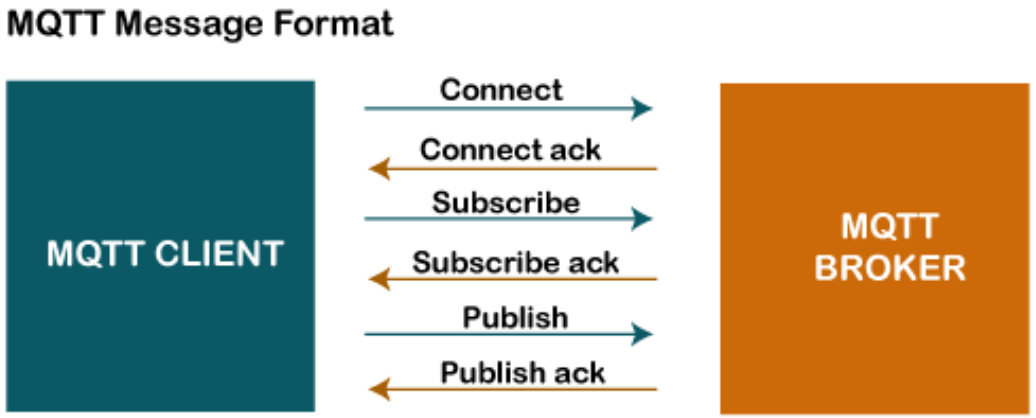
- Broker - dispozitivul sau un program care permite clientului să publice mesajele și să se aboneze la mesaje
- Acțiuni îndeplinite de către broker:
  - acceptă conexiunea la rețea de la client
  - acceptă mesajele de la client
  - procesează cererile de abonare și dezabonare
  - transmite mesajele aplicației către client
  - închide conexiunea de rețea de la client
- Când un broker și un client abonat pierd contactul, brokerul va stoca mesajele într-un buffer și le va trimite ulterior

Topic

- Clienții publică către brokeri mesaje pe diferite subiecte (*topic*)
- Brokerul este serverul central care primește aceste mesaje și le filtrează în funcție de topic
- Mesaje sunt trimise doar acelor clienți care s-au abonat la topic-ul respectiv

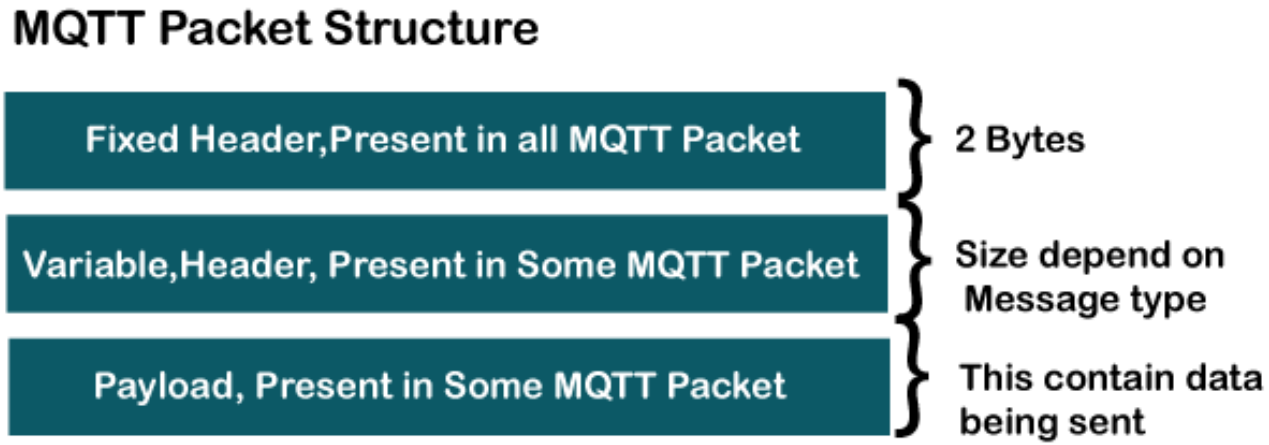
Format Mesaj

- MQTT utilizează comanda și formatul de confirmare a comenzii
- Fiecare comandă are asociată o confirmare (*ack*)
  - comanda de conectare are confirmarea de conectare
  - comanda subscribe are confirmarea de abonare
  - comanda de publicare are confirmarea de publicare



Format Mesaj

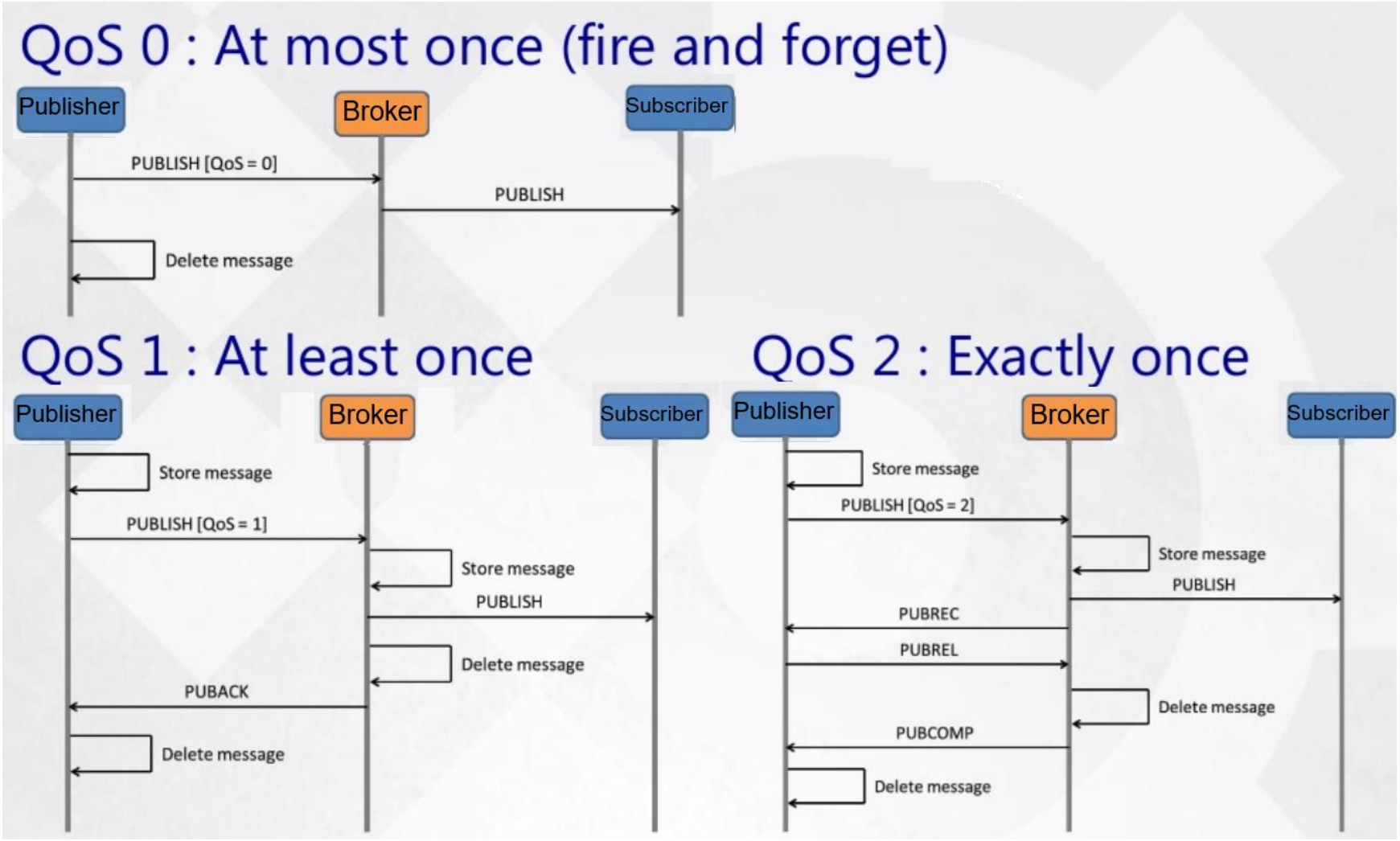
- Formatul mesajului MQTT constă din:
  - **Antet fix** de 2 octeți, prezent în toate pachetele MQTT
  - **Antet variabil,** care nu este întotdeauna prezent
  - Al treilea câmp este **payload-ul**, care este, de asemenea, opțional. Acesta conține practic datele care sunt trimise



Caliteatea Serviciului

- Fiecare conexiune la broker poate specifica o măsură de tip Quality of Service (QoS):
  - **At most once**– mesajul este trimis o singură dată, iar clientul și brokerul nu iau măsuri suplimentare pentru a confirma livrarea
  - **At least once** – mesajul este reîncercat de către expeditor de mai multe ori până când este primită confirmarea
  - **Exactly once** – expeditorul și destinatarul se angajează prin *handshake* că este primită o singură copie a mesajului

QoS


QoS 0 : At most once (fire and forget)
Publisher
Broker
Subscriber
PUBLISH [QoS = 0]
PUBLISH
Delete message
QoS 1 : At least once
Publisher
Broker
Subscriber
Store message
PUBLISH [QoS = 1]
Store message
PUBLISH
Delete message
PUBACK
Delete message
QoS 2 : Exactly once
Publisher
Broker
Subscriber
Store message
PUBLISH [QoS = 2]
Store message
PUBLISH
PUBREC
PUBREL
Delete message
PUBCOMP
Delete message

Avantaje

- Protocol *lightweight*, ce permite transportul eficient de date
- Utilizare minimă a pachetelor de date
- Livrare promptă și eficientă a mesajelor
- Minimizează consumul de energie
- Transmiterea datelor este rapidă, deoarece mesajele MQTT au o amprentă mică de cod
- Aceste mesaje de control au un antet fix de dimensiunea de 2B și un mesaj de încărcare utilă până la dimensiunea de 256 MB

Dezavantaje

- În comparație cu Protocolul de aplicare constrâns (CoAP), MQTT are cicluri de trimitere mai lente
- Descoperirea resurselor în MQTT se bazează pe un abonament flexibil la topic, în timp ce descoperirea resurselor în CoAP se bazează pe un sistem de încredere
- MQTT nu oferă posibilitate de criptare a datelor. Mai degrabă, criptarea de securitate este realizată prin TLS/SSL
- Construirea unei rețele MQTT scalabile la nivel internațional este dificil de realizat

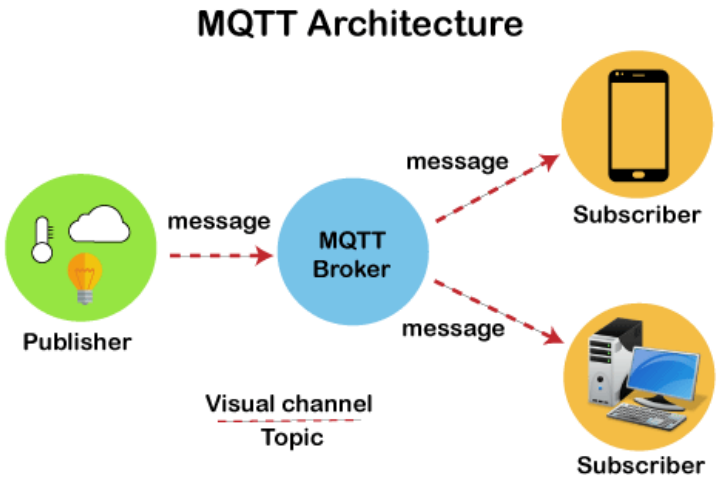
MQTT vs HTTP

	3G		Wifi	
	HTTPS	MQTT	HTTPS	MQTT
% Battery / Hour	18.43%	**16.13%**	**3.45%**	4.23%
Messages / Hour	1708	**160278**	3628	**263314**
% Battery / Message	0.01709	**0.00010**	0.00095	**0.00002**
Messages Received	240 / 1024	**1024 / 1024**	524 / 1024	**1024 / 1024**

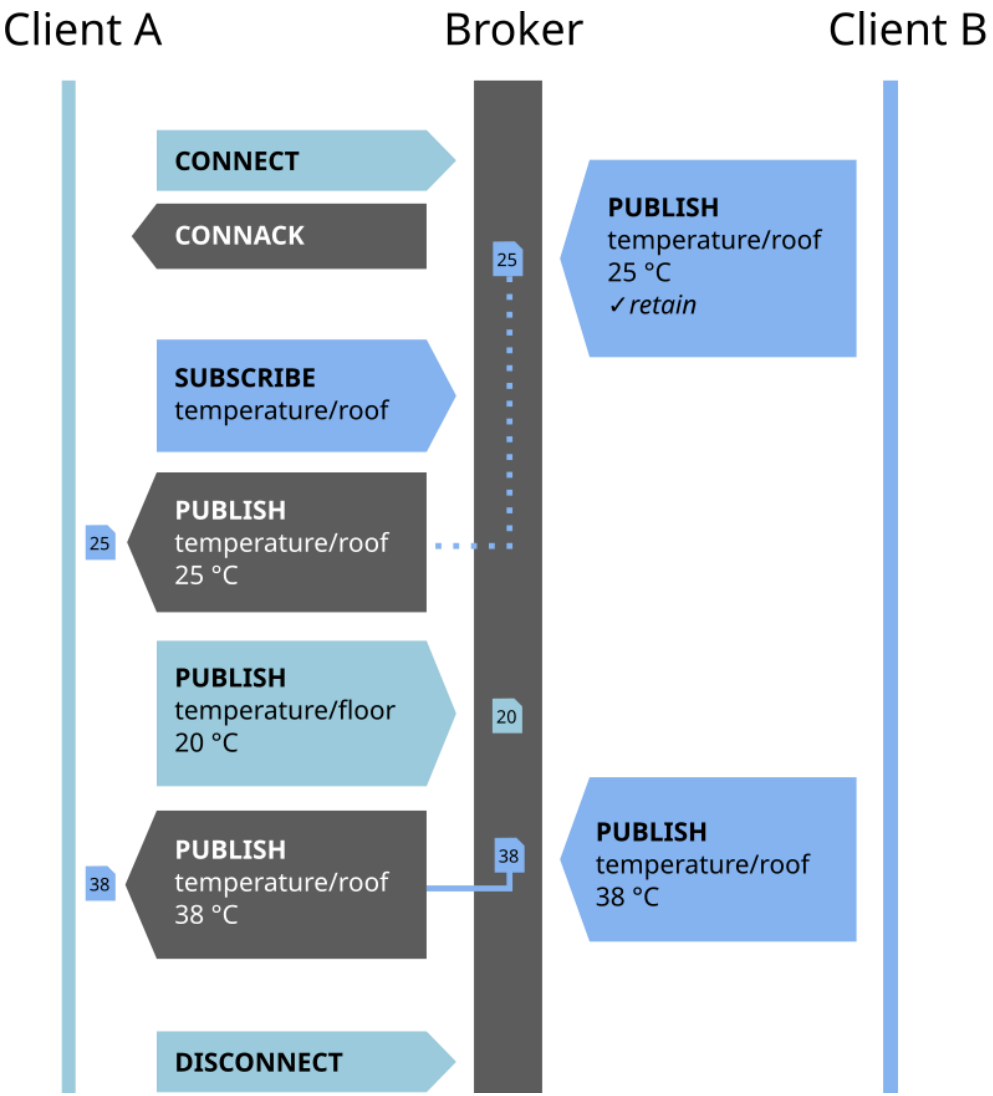
Sursă: https://stephendnicholas.com/posts/power-profiling-mqtt-vs-https

Exemplu MQTT

- Dispozitiv ce are un senzor de temperatură și trimite valoarea către server (broker)
- Dacă un telefon/PC dorește să primească această valoare a temperaturii:
  - Publisher-ul definește mai întâi un topic (temperatura) și publică mesajul (valoarea temperaturii)
  - Telefonul/PC-ul se va abona la acel topic și va primi mesajul publicat, adică valoarea temperaturii



Exemplu MQTT



Implementare HTTP/MQTT

Server HTTP

Publisher/Subscriber MQTT

ÎNTREBĂRI ?